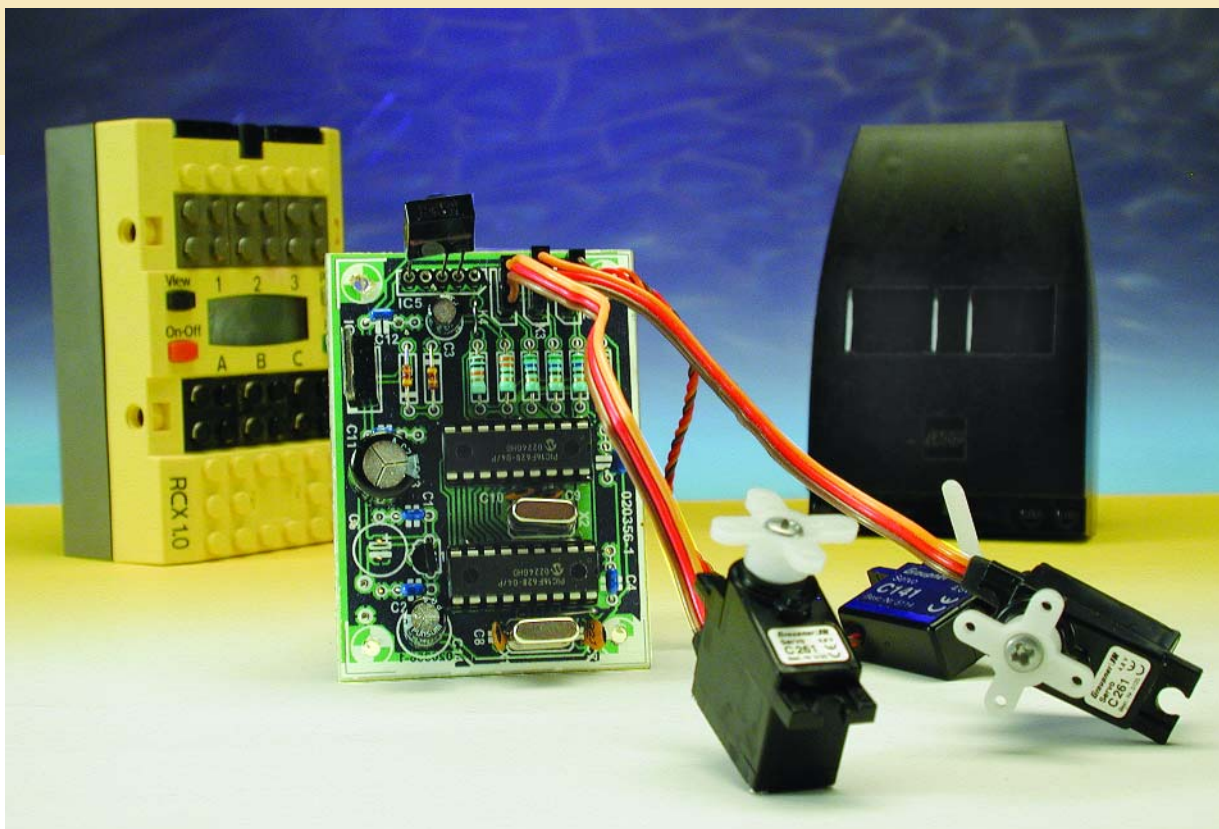




IR Servo Motor Interface for RCX

LEGO RCX-compatible infrared remote control for servos

One of the few features lacking from the LEGO Mindstorms system is an accurate positional drive. This has prompted the author to develop this interface circuit, which can control up to three servos of the type used in radio-controlled models via the IR-interface of the RCX brick.

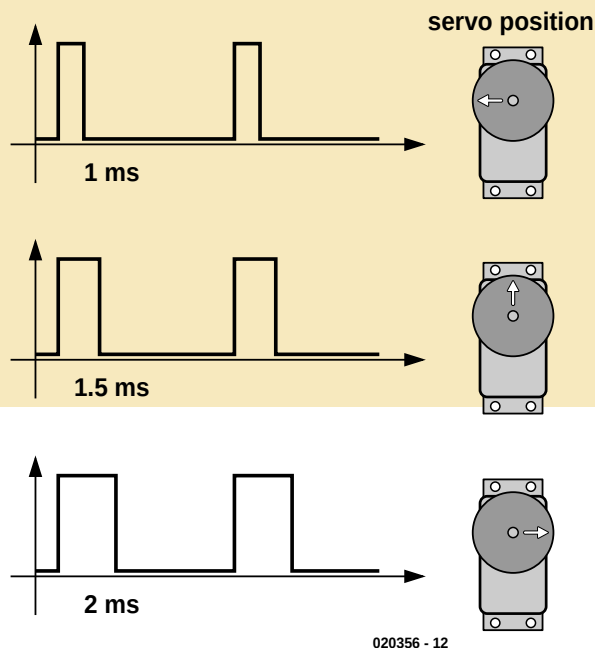


Figure 1. Driving a servo using a rectangular waveform whose pulse width determines the servo's position.

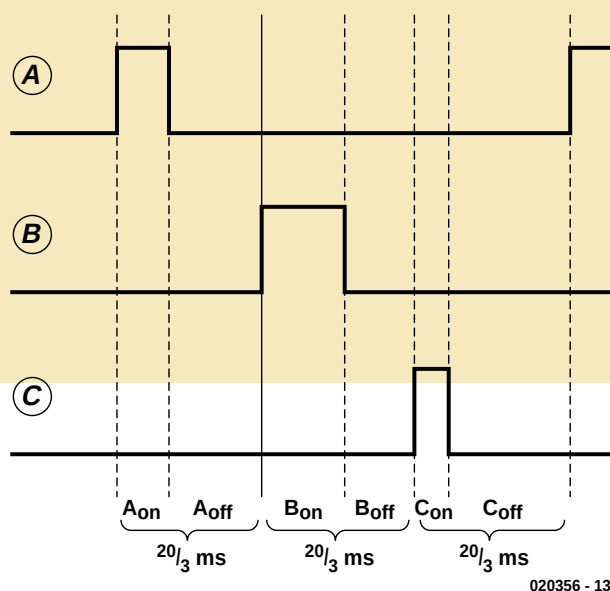


Figure 2. Timing diagram showing the output signals of the PIC to control three servos.

The three different types of LEGO Technic 9 V motors are more than adequate for most robotics projects. However, when accurate angular positioning is required, one can quickly become frustrated by the amount of play in the gears and start looking for a better solution.

The servos used in radio-controlled models offer more precision. Of course, we will not only have to deal with mechanical interface problems of joining to other LEGO bricks: we will also have to operate under control of the RCX, which is a little trickier. The RCX brick is the main control element in the LEGO Mindstorms system. It consists of a microcontroller built into a large, bright yellow, LEGO brick.

Ralph Hempel, the creator of pbForth, has, along with Andreas Peter, developed various add-on circuits that allow the RCX to be used with two servo motors. Descriptions of the circuits are available on the Internet: see references [1] and [2]. Hempel has provided special program commands in pbForth to drive his circuit, namely `SERVO_INIT` and `SERVO_SET` [3].

The circuit presented here will allow you to control up to three servos simultaneously with the RCX standard version 2.0 firmware, using the IR interface on the RCX brick.

Servo control

Servos for radio-controlled models use a form of pulse-width modulation (PWM). The control signal is a rectangular waveform with a frequency of about 50 Hz. The precise frequency is not critical: it need only lie somewhere in the range 30 Hz to 60 Hz. The information is carried only in the width of the rectangular pulse, which is allowed to vary between 1 ms and 2 ms. As **Figure 1** shows, the servo is in its mid-position when the pulse width is 1.5 ms. Inside the servo there is a potentiometer attached to the output drive shaft. This gives the electronics in the servo the necessary angle feedback information. Because of the potentiometer end-stops, the servo should never be overdriven, or it will be permanently damaged. This can happen if the pulse width is outside the permitted range.

In order to drive three servos at the same time, three rectangular waveforms need to be generated whose pulse widths can be adjusted independently of one another. As can be seen from the circuit diagram in **Figure 3**, a microcontroller type PIC16F628 (IC3) generates these signals. The program in the PIC is organised in such a way that the three control pulses are produced one after another in sequence at an overall rate of 50 Hz. Thus we have:

$$A_{on} + A_{off} + B_{on} + B_{off} + C_{on} + C_{off} = 1/50 \text{ s} = 20 \text{ ms}$$

We are given A_{on} , B_{on} and C_{on} , the desired pulse widths. The 20 ms total time available is divided into three equal parts, giving:

$$A_{off} = 20/3 \text{ ms} - A_{on}; B_{off} = 20/3 \text{ ms} - B_{on}; C_{off} = 20/3 \text{ ms} - C_{on}$$

The behaviour is illustrated in the diagram in **Figure 2**. The pulse generator program operates as follows (in much-simplified pseudocode):

```
Label_1
  OutA on
  Wait A_on
  OutA off
  Wait A_off

  OutB on
  Wait B_on
  OutB off
  Wait B_off

  OutC on
  Wait C_on
  OutC off
  Wait C_off

  Goto Label_1
```

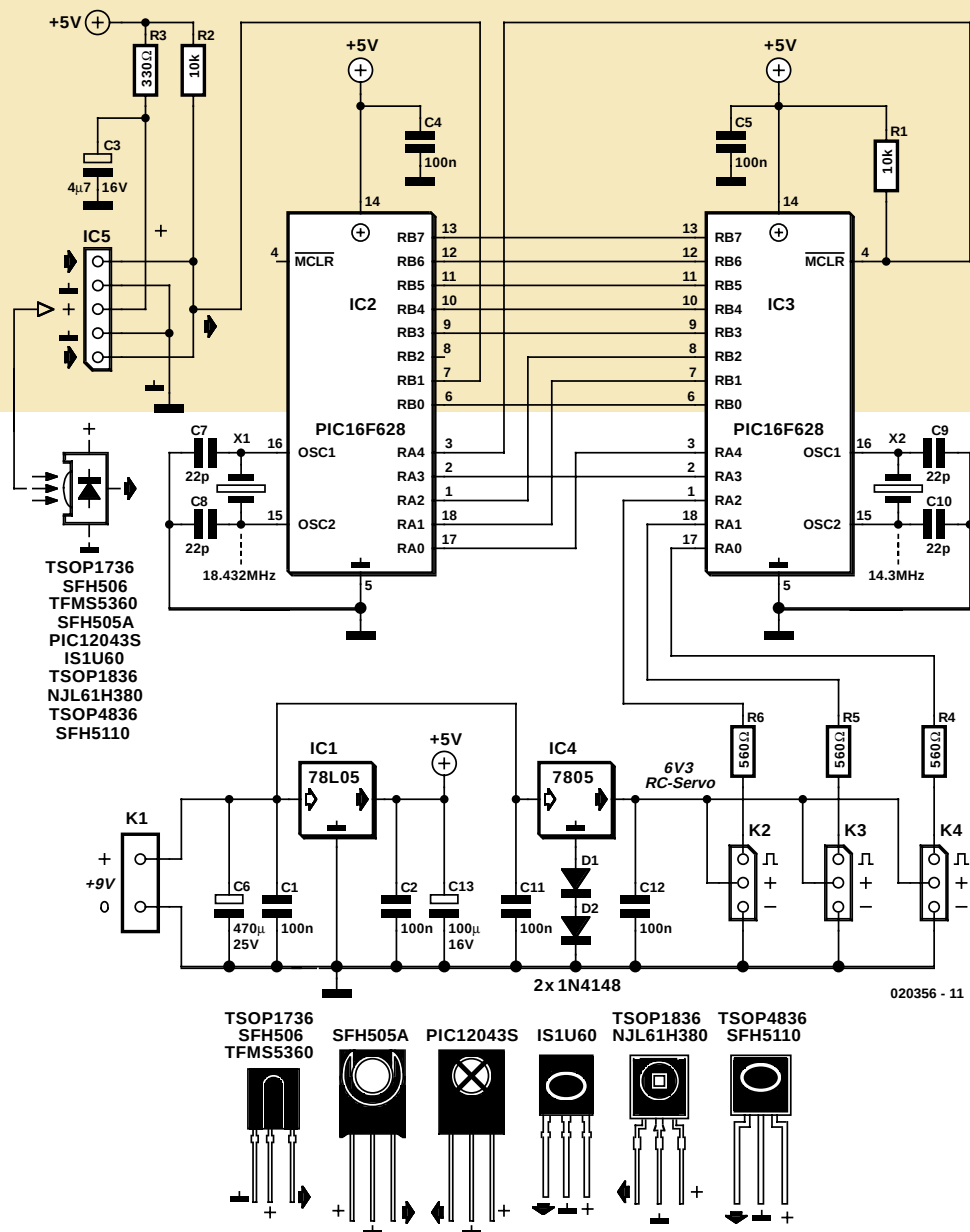


Figure 3. There are three main parts to the circuit: the dual regulated power supply (IC1 and IC4), the IR receiver (TSOP1738 and IC2), and the pulse generator (IC3).

Interface Circuit

The circuit in **Figure 3** is designed to operate from a separate 9 V battery supply. The two voltage regulator circuits using separate voltage regulator ICs (IC1 and IC2) mean that the main supply voltage does not drop when the servos move, ensuring trouble-free operation. If all three motors are moving at the same time, the current draw can be up to around 500 mA. The interface should therefore under no circumstances be connected to an RCX output, which is not suitable for such currents. Note also the rather odd use of diodes D1 and D2. A voltage of about 0.65 V is dropped across each diode, which shifts the voltage at the

ground pin of the 7805 by 1.3 V. This increases the output voltage of regulator IC4 to 6.3 V, which is the ideal voltage for the servos.

The actual IR servo motor controller circuit consists of IR receiver IC5 and two type PIC16F628 microcontrollers connected to one another. The first microcontroller (IC2) converts the serial data obtained from the IR receiver to parallel, and sends it on to the second microcontroller (IC3). This is done using an 8-bit data bus and three control signals. The maximum resolution achievable is one part in 256. The 1 ms time period is divided into 255 steps which, depending on the characteristics of the servo, gives an angular

resolution of less than 1°. On some types of servo motor this resolution is greater than the so-called 'dead band' (typically 5 µs), and so it is possible that the servo motor will not react to a change of just one unit in the control value. (The 'dead band' is a necessary feature of servo motors to ensure that while the motor is turning, the incoming control signal and the internal reference signal can be aligned as quickly as possible, but without having the rotor oscillate about the target position.)

The second PIC (IC3) operates at a clock frequency of 14.3 MHz, a value chosen on the basis of the relationship between the servo control pulse

widths and the PIC instruction cycles. The clock period is 69.9 ns: as in the earlier PIC16F84 device, most of the RISC instructions of the PIC16F628 are executed in 4 clocks, and this is also the rate at which the timer increments. An instruction therefore takes 279.9 ns. The program in IC3 uses the 16-bit timer (TMR1), and the shortest servo pulse of 1 ms corresponds to 3575 increments of TMR1. This value is also the one used on initialisation. There follows an 'off' period of (20/3 – 1) ms, which corresponds to 20258 increments. The pulse length in PIC cycles is calculated as follows:

$t = a * x + b$, where:

$0 < x \leq 255$
(0 is a reserved value)

$a = (7150 - 3575) / (255 - 1)$
 $= 14.0$ (slope)

$b = 3575$ (intercept)

Now we can see why the particular clock frequency of IC3 was chosen: the quotient 3575/254 is approximately an integer. The program in the PIC has the facility to use slope and intercept values different from those given above in order to allow the servos to be trimmed. There are certain limitations in the communications protocol which mean that we can only send 8-bit values: this means that the slope can only be set to integer values, and the intercept can only be adjusted in steps of 20.

Data transfer

Parallel data transfer between IC2 and IC3 takes place in two stages. Two microcontrollers are used because there are several time-critical operations to be carried out. First,

Table 1 Control lines truth table

IC2 (UART)					
A4	A3	A0	Value	PortA Value (A7-A5, A2, A1 masked)	Meaning
0	0	0	0	0	NOP
0	0	1	1	1	Load A
0	1	0	2	8	Load B
0	1	1	3	9	Set Slope
1	0	0	4	16	Charge C
1	0	1	5	17	Set Intercept
1	1	0	6	24	NOP
1	1	1	7	25	Reset

Table 2

Additional LEGO protocol data to monitor transmission	
Data header (for all opcodes)	hex 55 FF 00
Opcode	0x14 (=d20)
Opcode Complement	0xFF-0x14 = 0xEB
Dummy	0x00
Dummy Complement	0xFF
Dummy	0x00
Dummy Complement	0xFF
Instruction	n
Instruction Complement	not(n)
Data	x
Data Complement	not(x)
Checksum	s
Checksum Complement	not(s)

Table 3

ROBOLAB Command overview (Device Instruction Set)		
Instruction	Number	Command
1 (1)	0	Select channel A
2 (8)	0	Select channel B
4 (16)	0	Select channel C
1 (1)	1..255	Charge A
2 (8)	1..255	Charge B
4 (16)	1..255	Charge C
3 (9)	0..255	Set slope of selected channel (default 14)
5 (17)	0..5100 (0..255)	Set intercept (DIV 20) of selected channel (default 3575)
7 (25)	x	Reset all slope and intercept values to default

the UART (Universal Asynchronous Receiver and Transmitter) must always be ready to receive data, since the interface does not know when the next IR command will arrive. Second, the UART must manage reception of data according to the LEGO protocol (opcode 0x14), which uses a block size of 15 bytes at 2400 baud (using one start bit, 8 data bits, a parity bit and one stop bit): this gives a total transmission time of about 70 ms per command. Finally, three independent stable

pulse-width modulated signals must be generated according to the incoming commands, and the jitter in these pulses must be kept within the dead band of the servo. The effort involved in trying to integrate all these functions into a single reliable program in one PIC makes the price of a second PIC16F628 look insignificant.

Pins RB1 and RB2 of IC2 are configured for use with the UART and so are not available to form part of the parallel data bus. RA1 and RA2 are

COMPONENTS LIST

Resistors:

R1,R2 = 10k Ω
R3 = 330 Ω
R4,R5,R6 = 560 Ω

Capacitors:

C1,C2,C4,C5,C11,C12 = 100nF
C3 = 4 μ F 16V radial
C6 = 470 μ F 16V radial
C7-C10 = 22pF
C13 = 100 μ F 16V radial

Semiconductors:

D1,D2 = 1N4148
IC1 = 78L05
IC2,IC3 = PIC16F628, programmed,
order codes 020356-41 (IC2),
020356-42 (IC3)
IC4 = 7805
IC5 = TSOP1738 (see text)

Miscellaneous:

K1 = 9-V battery (see text)

K2,K3,K4 = 3-way pinheader

X1 = 18.432MHz quartz crystal

X2 = 14.3MHz quartz crystal (see text)

PCB, order code 020356-1 (see Readers
Services page)

Disk, order code 020356-11 or Free
Download

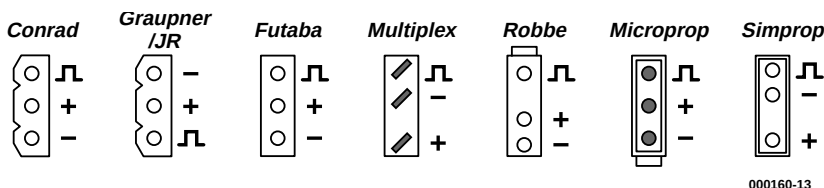


Figure 5. Pinouts of various commonly found radio control servos.

used instead. This arrangement requires the use of at least one control signal, since the two ports of IC2 cannot be updated simultaneously. Without a control signal, IC3 could read erroneous values between the updates of the ports. In looking at the control signals, note that RA4 of IC2 is an open-drain output and therefore requires a 1 k Ω pull-up resistor (R1).

The two ICs are configured so that the internal $\overline{MCLR}$ signal is pulled to V_{DD} , thus freeing up the pin to be used as the RA5 input on IC3. **Table 1** shows the values on the control signal bus and what they signify.

RCX UART

Using the LEGO Mindstorms SDK2 [4] and version 2.0 of the firmware, the user has access to the transmit channel of the RCX UART. This opens the door to a new world of robotics projects. For example, an RCX can program another RCX, or several RCX bricks can jointly process sensor and variable data, simply by obeying the LEGO UART protocol and sending the right opcodes. Be warned, however, that communication will be with all RCX bricks in range of the IR commands. This can, according to the SDK, have 'catastrophic' results. For good reasons we have chosen the opcode 0x14 ('SETVAR' or 'SETV') for use here. This command is no longer

used in ROBOLAB 2.5, since a universal 'Set' command was introduced in version 2.0 of the firmware, which allows — depending on the configuration — both sensor values and other variables to be set.

According to the datasheet the full command is as follows:

SETVAR: 0x14 variable, source, number (LO), number (HI)

(Reply 0xE3. Note that once the receiving RCX brick has verified the opcode it sends a confirmation of reception. In our case this is not relevant, but a short pause should be inserted between consecutive opcode transmissions to avoid IR interference.)

We slightly modify this command:

0x14 0 (dummy), 0 (dummy), instruction, data

(The RCXs involved will unfortunately lose the use of control variable 0.)

The LEGO protocol employs several additional data bytes to confirm correct data transmission, as shown in **Table 2**. The receiver can check the complement bytes and compute the checksum over the received bytes: this can be compared with the received checksum. Overall, this gives a good check on the communication quality. Although the RCX carries out a parity check, it is not essential in the program in IC2 and

in any case the 16F628 UART does not provide an automatic parity check facility. The author used the low-cost TSOP1738 from Vishay-Telefunken as IR receiver and demodulator. LEGO IR transmissions are at 2400 baud and use a carrier of 38 kHz. This sensitive IR receiver gives a range of up to 10 m, when the RCX is configured for long-distance IR communications.

Table 3 gives an overview of the instructions used in the ROBOLAB program, with actual values sent given in brackets. As already mentioned, care should be exercised when trimming the servo parameters: in order to avoid overdriving the servo, with possible consequent damage, make adjustments only in small steps.

Construction

The printed circuit board (**Figure 4**) is small but easy to populate, and so construction should not present any problems. Care should be taken with the wire links: they are the price that must be paid for using a single-sided circuit board. There are six wire links in total, of which three are under the ICs, one being under IC2 and two under IC3. These wire links are best made with insulated wire on the underside of the board. Alternatively the links can be made on the component side using thin copper wire, passing under the ICs (or their sockets). In any case the wire links should be fitted first, followed by the passive components, and finally the active components. As always, take care with the soldering and pay attention to the component values, the polarity of the electrolytic capacitors and diodes, and the orientation of the ICs. The heatsink for the voltage regulator IC4 is mounted just off the edge of the board.

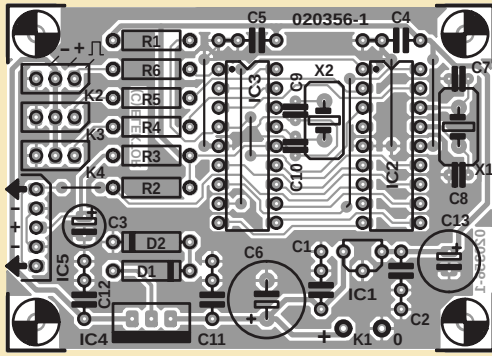


Figure 4. The single-sided circuit board is compact: note the wire links!

The 9 V battery is connected to the terminal pins marked K1. The current consumption of the circuit chiefly depends on the number and consumption of the servos. If space permits, it is best to use a battery made up from six AA-size cells: a 9 V PP3-type battery will have a much shorter life, perhaps by a factor of between five and ten.

IR receiver IC5 need not necessarily be soldered directly to the printed circuit board. It can be mounted remotely using a three-core cable running from the points on the board where IC5 would be located. Other IR receiver devices with similar characteristics can be substituted for the Vishay-Telefunken TSOP1738 specified in the parts list. Some of the alternatives available are shown in the circuit diagram (Figure 4) along with their pinouts. All the ICs have three pins, but on the printed circuit board five holes are provided for IC5, arranged so as to allow any of the suggested ICs to be directly soldered in (or wired) using three adjacent holes.

There is also an alternative to the 14.3 MHz crystal X2. If a crystal with this frequency is not available, a more readily-available 16 MHz part can be used instead. Ordinary radio control servos have no difficulty in coping with the effect of this change.

The three servos are connected to the headers at K2, K3 and K4. Again, attention must be paid to the pinout, and the various manufacturers of servos use different plugs on the end of the connection cable. The most frequently encountered arrangements are shown in **Figure 5**.

Operation

The servo driver interface can be controlled using ROBOLAB 2.5 with RCX firmware version 2.0, using the LEGO

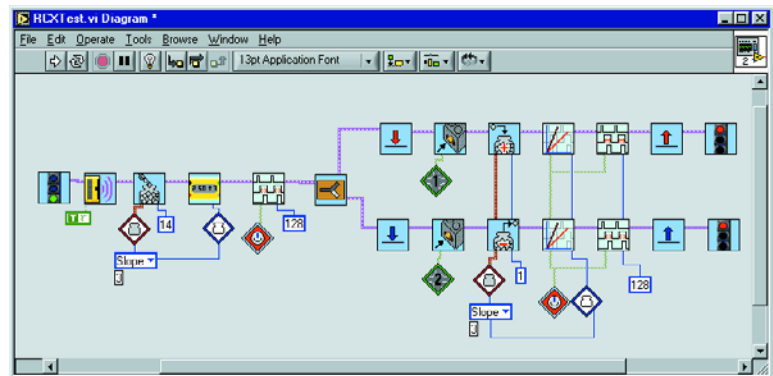


Figure 6. One of the RCX programs available for download from www.elektor-electronics.co.uk. The touch sensors can be used to adjust the servo's position.

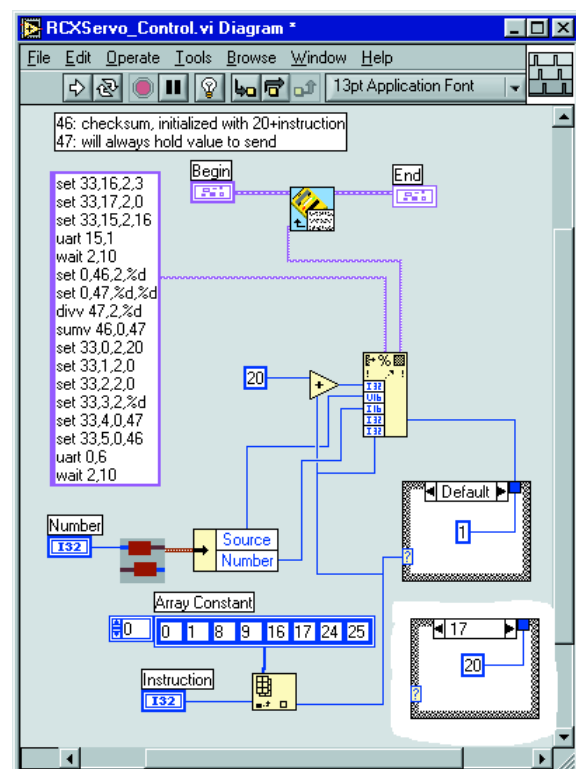


Figure 7. The heart of the RCX program: driving the RCX UART transmitter.

Web references

- [1] www.inchlab.net/2servo_interface.htm
- [2] <http://mindstorms.lego.com/sdk2/default.asp>
- [3] Dave Baum, Michael Gasperi, Ralph Hempel, Luis Villa:
'Extreme Mindstorms, An Advanced Guide to LEGO Mindstorms', p. 313.
- [4] <http://www.ceeo.tufts.edu/robola->

batceeo/Resources/documentation/
LEGOMindStormsSDK.zip

Free Downloads

PCB layout in PDF format. File number:
020356-1.zip

RCX program to drive the servos;
source and object code for the two

PICs. File number **020356-11.zip**
www.elektor-electronics.co.uk/dl/dl.htm, select month of publication.

Author's full documentation including pictures, programs, source and object code and instructions: http://www.con-vict.lu/IR_Servo_control.zip

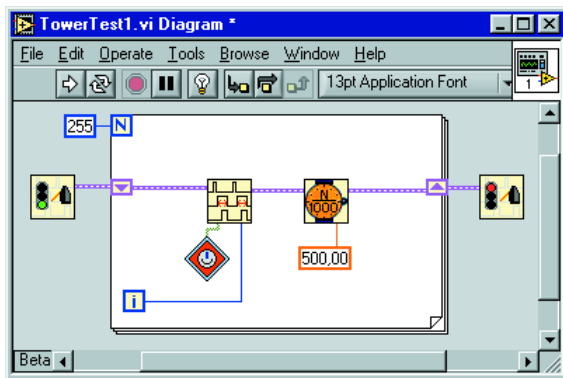


Figure 8. This simple program runs on the PC and controls the servos via the LEGO tower.

tower (RS232 or USB version), or from the RCX itself. Various programs are available for download (see the text box for addresses), of which we only have space here for a brief description.

Figure 6 shows a simple downloadable program with two parallel tasks. In this example the RCX increases or decreases the angular

position of the red servo, starting at its mid-position, according to which touch sensor is pressed. The colours red, blue and yellow follow the usual ROBOLAB colour coding convention. The second program icon puts the RCX into long-distance IR communications mode, and the display shows the current

slope value. The intercept value can also be trimmed in a similar fashion. Changing the slope increases or reduces the total range of motion of the servo, while changing the intercept only affects the zero position.

As usual in ROBOLAB/LabVIEW, entire program segments can be collected together and represented graphically by a single new icon, making it possible to take a high-level view of even the most complex programs. **Figure 7** shows the heart of the RCX program: driving the RCX UART transmitter. Further information on the use of the UART can be found in the LEGO Mindstorms SDK [4].

The simple program in **Figure 8** runs on the PC. The red servo motor turns slowly over its entire operating range. Note the use of a LabVIEW FOR loop with a shift register. Data leaving the main box on the right-hand side are stored in the shift register at the end of each iteration, and recirculated. This makes it very easy, for example, to check the program for any errors. The two icons outside the main box are responsible for tasks including initialising and shutting down communications with the LEGO tower. The program looks for the tower connection configured in ROBOLAB. The normal ROBOLAB RCX direct mode cannot be used, since the program would then wait forever for an acknowledgement signal from the RCX: in our case transmissions are in one direction only.

Finally, **Figure 9** shows the front panel of the RCServo Manager program, which is very easy to use. Both slope and intercept can be trimmed, and the tower only transmits when a value is changed. If a value is changed continuously, the value of RATE determines the transmission frequency.

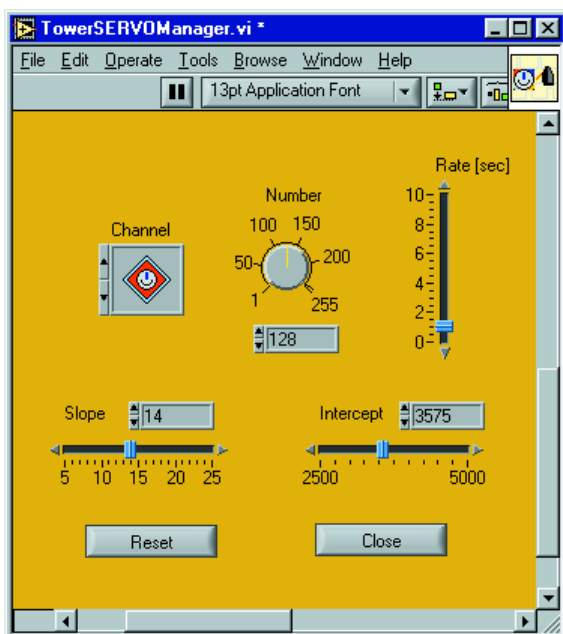


Figure 9. The easy-to-use front panel of the RCServo Manager.

(020356-1)